

WHAT IS CLAIMED:

1 1. A method of optimizing an intermediate representation of program code
2 used during the translation of the program code, comprising:
3 identifying register definitions in the intermediate representation;
4 determining whether a top-level expression for an identified register definition is
5 a byteswap operation; and
6 applying a lazy byteswapping optimization algorithm to delay performance of
7 said byteswap operation on a value until a byteswapped value is actually required.

1 2. The method of claim 1, wherein the lazy byteswapping optimization
2 algorithm comprises:
3 if said top-level expression is a byteswap operation, modifying the intermediate
4 representation by:
5 removing said byteswap operation as the top-level expression for said
6 identified register definition, and
7 wherever else in the intermediate representation where the register defined
8 by the identified register definition is referenced, modifying the
9 intermediate representation by inserting a byteswap operation above said
10 referenced register in the intermediate representation;
11 determining whether consecutive byteswap operations are present in the modified
12 intermediate representation; and
13 preventing byteswap operations appearing in the modified intermediate
14 representation from being performed.

1 3. The method of claim 2, wherein consecutive byteswap operations are
2 prevented from being performed by removing said consecutive byteswap operations from
3 the modified intermediate representation.

1 4. The method of claim 2, further comprising, whenever said byteswap
2 operation is removed as the top-level expression for said register definition, setting a lazy
3 byteswap flag for said register definition to indicate that the value contained in the
4 register definition is in an opposite byte order as expected.

1 5. The method of claim 4, wherein the intermediate representation modifying
2 step is performed wherever else in the intermediate representation where a register having
3 a register definition with a lazy byteswap flag that has been set is referenced.

1 6. The method of claim 4, further comprising clearing the set lazy byteswap
2 flag for said referenced register when a new value is stored in said register.

1 7. The method of claim 6, wherein a lazy byteswap state exists that includes
2 a set of all lazy byteswap flags for each of the registers in the intermediate representation,
3 further wherein each of said registers includes a respective lazy byteswap flag that
4 is either in a set or cleared state to indicate the current state of that register.

1 8. The method of claim 7, further comprising a step of synchronizing the
2 lazy byteswap state of the registers between blocks of program code being translated.

1 9. A computer-readable storage medium having software resident thereon in
2 the form of computer-readable code executable by a computer to perform the following
3 steps to optimize an intermediate representation of program code used during the
4 translation of the program code:

5 identifying register definitions in the intermediate representation;
6 determining whether a top-level expression for an identified register definition is
7 a byteswap operation; and
8 applying a lazy byteswapping optimization algorithm to delay performance of
9 said byteswap operation on a value until a byteswapped value is actually required.

1 10. The computer-readable storage medium of claim 9, wherein the lazy
2 byteswapping optimization algorithm comprises:
3 if said top-level expression is a byteswap operation, modifying the intermediate
4 representation by:
5 removing said byteswap operation as the top-level expression for said
6 identified register definition, and
7 wherever else in the intermediate representation where the register defined
8 by the identified register definition is referenced, modifying the
9 intermediate representation by inserting a byteswap operation above said
10 referenced register in the intermediate representation;
11 determining whether consecutive byteswap operations are present in the modified
12 intermediate representation; and
13 preventing byteswap operations appearing in the modified intermediate
14 representation from being performed.

1 11. The computer-readable storage medium of claim 10, wherein consecutive
2 byteswap operations are prevented from being performed by removing said consecutive
3 byteswap operations from the modified intermediate representation.

1 12. The computer-readable storage medium of claim 10, said computer-
2 readable code further executable for:
3 whenever said byteswap operation is removed as the top-level expression for said
4 register definition, setting a lazy byteswap flag for said register definition to indicate that
5 the value contained in the register definition is in an opposite byte order as expected.

1 13. The computer-readable storage medium of claim 12, wherein the
2 intermediate representation modifying step is performed wherever else in the
3 intermediate representation where a register having a register definition with a lazy
4 byteswap flag that has been set is referenced.

1 14. The computer-readable storage medium of claim 12, said computer-
2 readable code further executable for clearing the set lazy byteswap flag for said
3 referenced register when a new value is stored in said register.

1 15. The computer-readable storage medium of claim 14, wherein a lazy
2 byteswap state exists that includes a set of all lazy byteswap flags for each of the registers
3 in the intermediate representation,
4 further wherein each of said registers includes a respective lazy byteswap flag that
5 is either in a set or cleared state to indicate the current state of that register.

1 16. The computer-readable storage medium of claim 15, said computer-
2 readable code further executable for synchronizing the lazy byteswap state of the
3 registers between blocks of program code being translated.

1 17. An apparatus for use in a computing environment having a processor and a
2 memory coupled to the processor for optimizing an intermediate representation of
3 program code used during the translation of the program code, said apparatus comprising:
4 a register identifying mechanism for identifying register definitions in the
5 intermediate representation;
6 a byteswap determining mechanism for determining whether a top-level
7 expression for an identified register definition is a byteswap operation; and
8 a lazy byteswapping mechanism for applying a lazy byteswapping optimization
9 algorithm to delay performance of said byteswap operation on a value until a
10 byteswapped value is actually required.

1 18. The apparatus of claim 17, wherein the lazy byteswapping mechanism is
2 further configured for:
3 if said top-level expression is a byteswap operation, modifying the intermediate
4 representation by:

5 removing said byteswap operation as the top-level expression for said
6 identified register definition, and
7 wherever else in the intermediate representation where the register defined
8 by the identified register definition is referenced, modifying the
9 intermediate representation by inserting a byteswap operation above said
10 referenced register in the intermediate representation;
11 determining whether consecutive byteswap operations are present in the modified
12 intermediate representation; and
13 preventing byteswap operations appearing in the modified intermediate
14 representation from being performed.

1 19. The apparatus of claim 18, wherein consecutive byteswap operations are
2 prevented from being performed by the lazy byteswapping mechanism by removing said
3 consecutive byteswap operations from the modified intermediate representation.

1 20. The apparatus of claim 18, wherein the lazy byteswapping mechanism is
2 further configured for, whenever said byteswap operation is removed as the top-level
3 expression for said register definition, setting a lazy byteswap flag for said register
4 definition to indicate that the value contained in the register definition is in an opposite
5 byte order as expected.

1 21. The apparatus of claim 20, wherein the lazy byteswapping mechanism is
2 further configured such that the intermediate representation is modified wherever else in
3 the intermediate representation where a register having a register definition with a lazy
4 byteswap flag that has been set is referenced.

1 22. The apparatus of claim 20, wherein the lazy byteswapping mechanism is
2 further configured for clearing the set lazy byteswap flag for said referenced register
3 when a new value is stored in said register.

1 23. The apparatus of claim 22, wherein a lazy byteswap state exists that
2 includes a set of all lazy byteswap flags for each of the registers in the intermediate
3 representation,
4 further wherein each of said registers includes a respective lazy byteswap flag that
5 is either in a set or cleared state to indicate the current state of that register.

1 24. The apparatus of claim 23, further comprising a synchronizing mechanism
2 for synchronizing the lazy byteswap state of the registers between blocks of program
3 code being translated.